ISSN (Online) 2747-0563
Volume 2 Tahun 2021

Load Balancing Topologi Bipartite Pada Jaringan SDN

Henni Endah Wahanani[1], Mohammad Idhom[2], Eka Prakarsa Mandyartha [3]

[1,2,3] Program Studi Informatika, Universitas Pembangunan Nasional Veteran Jawa Timur
[1]henniendah.if@upnjatim.ac.id

***Abstrak*— Layanan teknologi telah berkembang dengan keandalan yang tinggi, oleh karena itu diperlukan sebuah konsep sistem kendali terpusat untuk mengatur perangkat jaringan pada sebuah infrastruktur jaringan yang disebut SDN (*Software Defined Network)*, dengan memisahkan antara sistem kontrol (*control plane*) dan sistem *forwarding* (*data plane*). Pengontrol dapat memberikan kontrol terpusat dengan menginstal aturan penerusan dalam bidang data, dan switch melakukan operasi yang berbeda pada paket sesuai dengan aturan ini. Cara komunikasi antara perangkat dan *controller* menggunakan sebuah protokol yang disebut dengan *Openflow*. Untuk mendukung SDN diperlukannya sebuah metode untuk mendistribusikan trafik jaringan komputer secara seimbang agar trafik jaringan komputer berjalan secara maksimal, metode itu adalah *load balancing*. Dalam penelitian ini melakukan ujicoba *load balancing* topologi bipartite di ujicoba pada 3 paket yaitu *UDP Flow*, *DNS*, dan *Telnet* dengan parameter yang diuji adalah *delay* dan *packet rate* dengan mengirimkan 1000 paket dengan ukuran setiap paket 100Kb selama 60s dengan *background* trafik setiap *link* 100 Mbit/s. Hasil dari pengujian *delay* yang terkecil terdapat pada paket DNS topologi 1 dengan 11,382 ms, dan *packet rate* terbesar pada paket *telnet* dengan 92,02 pkt/s.**

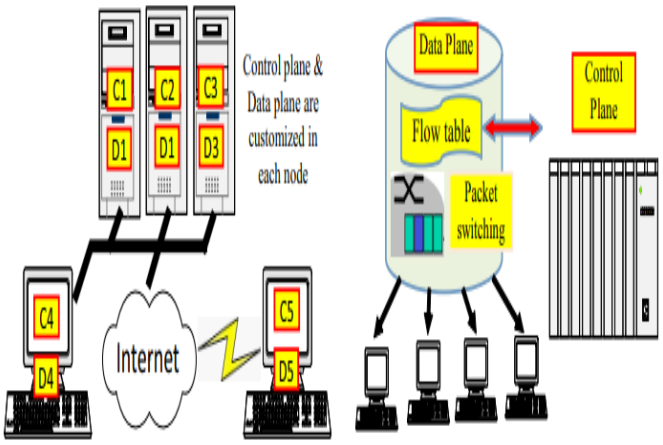
Kata Kunci— **load balancing, ECMP, bipartite, SDN**

I. PENDAHULUAN

Ketika saat pengguna yang banyak untuk memenuhi kebutuhan internet yang memiliki tersedianya dan keandalan yang tinggi, maka dibutuhkan jalur trafik jaringan yang merata agar tidak terjadi penumpukan pada jalur trafik jaringan tertentu saja. Untuk menghindari penumpukan pada jalur trafik jaringan tertentu saja, maka diperlukannya metode *load balancing*. *Load balancing* merupakan sebuah metode untuk mendistribusikan trafik jaringan secara seimbang agar trafik jaringan berjalan secara maksimal [1], selain itu *load balancing* merupakan teknik utama untuk meningkatkan kinerja dan skalabilitas internet. Dengan adanya metode *load balancing* maka setiap *gateway-gateway* yang terdapat pada sistem jaringan dapat digunakan secara maksimal. Dengan menggunakan metode *load balancing*, maka dapat memaksimalkan utilitas setiap *gateway-gateway* yang tersedia pada sebuah sistem jaringan.

Sebagian besar jaringan IP menjalankan *Protocol Gateway Interior* salah satunya yaitu OSPF (*Open Shortest Path First)* [2], untuk menghitung semua pasangan jalur terpendek antara router berdasarkan di konfigurasi statis (dimana bobot tersebut menentukan jaraknya dalam perhitungan jalur terpendek). Fitur ECMP (*Equal Cost Multipath*) adalah diperkenalkan untuk mengeksploitasi keragaman jalur terpendek dengan memungkinkan "pemisahan atau *split*" trafik antara beberapa jalur terpendek *hashing* statis per aliran [3][4]. Oleh karena itu, trafik jaringan saat ini membatasi trafik nya dalam 2 hal yaitu : 1. Trafik dari sumber ke tujuan dalam jaringan hanya dapat mengalir bersama jalur terpendek di antara jaringan dan 2. Trafik hanya dapat dibagi antara beberapa jalur terpendek (jika ada beberapa jalur terpendek) dengan cara yang sangat spesifik.

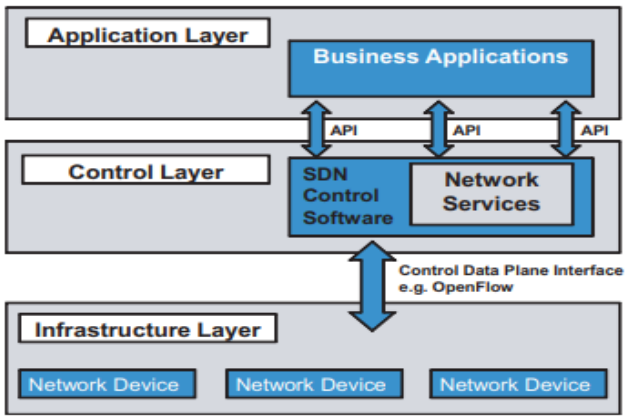
SDN menghasilkan peningkatan kinerja jaringan dalam hal manajemen jaringan, control dan penanganan data. SDN adalah solusi potensial untuk masalah yang dihadapi oleh jaringan konvensional [5][6][7] dan mendapatkan lebih banyak penerimaan di aplikasi seperti *cloud computing*. Hal ini dapat digunakan dalam data pusat dan sistem yang dioptimalkan beban kerja [8].



Gbr. 1. Perbandingan jaringan konvensional (kiri) dan SDN (kanan) [8]

Dengan menggunakan SDN, administrator memiliki kemampuan untuk mengontrol aliran data sebagai serta untuk mengubah karakteristik perangkat switching (atau perangkat routing) di jaringan dari lokasi pusat, dengan aplikasi control yang diimplementasikan sebagai modul perangkat lunak tanpa perlu berurusan dengan setiap perangkat secara individual. Ini memberi administrator jaringan kemampuan untuk mengubah tabel routing (jalur *routing*) secara sewenang-wenang di jaringan perangkat *routing*. Ini juga memungkinkan lapisan control ekstra atas data jaringan karena administrator dapat menetapkan tinggi atau rendah prioritas untuk paket data tertentu atau mengijinkan atau memblokir paket tertentu mengalir melalui jaringan. SDN memiliki tiga lapisan infrastruktur (perangkat jaringan), lapisan *control* (layanan jaringan) dan yang ketiga adalah lapisan aplikasi [8].

ISSN (Online) 2747-0563
Volume 2 Tahun 2021



Gbr. 2. Arsitektur SDN [8]

Control adalah bagian dari perangkat jaringan komputer yang akan menentukan bagaimana perangkat merespon aliran/paket/*frame* [9]. Kunci komponen dalam SDN adalah *controller* dimana ada POX [10], *OpenDaylight* [11], *Floodlight* [12], *Beacon* [13], *Ryu* [14], NOX [15], dan lain-lain adalah beberapa diantaranya dengan fitur berbeda yang dibandingkan oleh penulis [16]. Cara komunikasi antara perangkat dan *controller* menggunakan protokol yang di sebut *Openflow*. *Openflow* [17] adalah protokol standar komunikasi yang mampu membuat pertimbangan antara bagian *controller* dan bagian data perangkat jaringan, serta mampu menciptakan komunikasiyang sangat baik antara bagian kontrol dan bagian data. Dengan membuat bagian kontrol terpusat, ditetapkan SDN menyediakan jaringan yang lebih mudah di kelola dengan manajemen yang fleksibel, mudah di kelola dalam hal keamanan, optimalisasi sumber daya dan bahkan pengaturan jaringan dapat dilakukan sendiri tanpa menunggu perkembangan *vendor* untuk optimasi jaringan.
Penulisan dari naskah ini disusun sebagai berikut. Bagian II berfokus pada metodologi, Bagian III menyediakan hasil dan pembahasan, kemudian pada bagian IV kesimpulan.

II. METODOLOGI

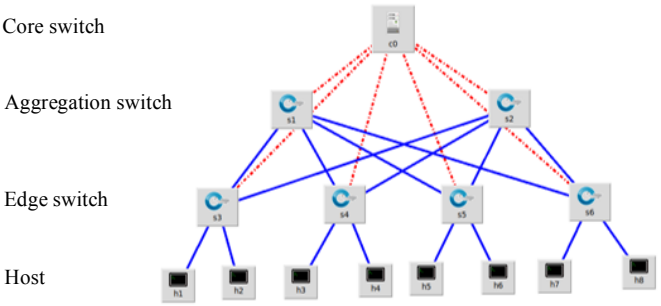
A. Rancangan Jaringan

Pada pembuatan sistem ini perangkat jaringan yang digunakan adalah *progammable switch* dengan menggunakan protokol Openflow 1.3. Jaringan *ethernet* digunakan untuk menghubungkan seluruh *switch* yang terdapat pada sistem. *Network Interface* pada setiap *switch* tidak memiliki IP *Address*, dikarenakan IP *Address gateway* dan *network interface* tersimpan di dalam *controller*. Aplikasi SDN pada sistem ini diterapkan pada *ryu controller* dan juga penerapan *load balancing* juga terdapat pada *controller*.
Host pada sistem ini dapat berupa komputer *client*, *server* maupun perangkat *end user* lainnya. *Host* akan mendapatkan IP *Address* dinamis yang diatur oleh DHCP (*Dynamic Host Configuration Protocol*) yang dijalankan oleh *controller*. *Switch* pada sistem ini memiliki posisi sebagai *relay* antara *host* dan *controller* dalam proses DHCP *discover-offer-request*. IP *Address* pada sistem ini hanya memiliki satu *subnet* dalam satu jaringan.

Untuk menjalankan *load balancing* pada arsitektur jaringan terdapat dua fungsi sistem yang dijalankan, yaitu sistem utama dan sistem pendukung. Sistem utama merupakan sistem yang menjalankan fungsi utama yaitu *load balancing*. Agar sistem utama dapat berjalan, maka diperlukan sistem pendukung yang dapat menjalankan fungsi-fungsi seperti proses *forwarding*, DHCP *service*, monitoring dan Proxy ARP.
Di dalam skenario uji coba untuk kebutuhan perangkat lunak di gunakan Ubuntu *desktop* digunakan sebagai sistem operasi *controller* dan juga sebagai sistem operasi *mininet*, *windows* 10 sebagai sistem operasi *laptop* yang nantinya akan dijadikan sebagai *laptop* virtualisasi, *ryu controller* sebagai aplikasi *control plane*, *Iperf* digunakan untuk aplikasi melihat trafik jaringan) dan D-ITG digunakan untuk aplikasi menghitung trafik jaringan.
Trafik uji : UDP Flow, DNS dan Telnet dengan parameter-parameter pengujian : *delay* dan *packet rate* dengan rate constant 1000 paket/s, setiap paket berisi 100KB dengan waktu selama 60s dengan *background traffic* sebesar 100Mbit/s.

B. Topologi Bipartite

Topologi dirancang dalam lingkungan virtual yang di bangun menggunakan aplikasi VirtualBox. Topologi jaringan yang digunakan adalah topologi *bipartite* karena memiliki keuntungan yaitu memiliki beberapa jalur yang digunakan untuk mengirimkan paket dari satu *host* ke *host* lainnya dan juga kalau terjadi kabel jaringan putus aktivitas transfer data masih bisa berjalan melalui jalur jaringan yang lainnya. *Switch* pada topologi *bipartite* dibagi menjadi 2 kelompok yaitu pada bagian atas adalah *aggregation switch* berguna untuk menggabungkan beberapa port ethernet menjadi *single link* secara *logic* dan *fast forwarding* melalui *hardware* sedangkan bagian bawah adalah *Edge switch* yang hanya dapat bekerja pada MAC *ethernet*. Gbr. 1 menunjukkan topologi 1 yang digunakan.

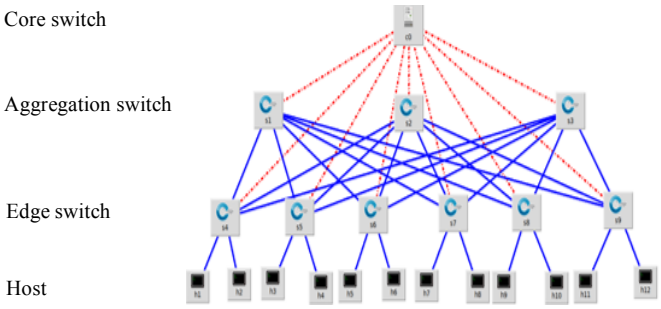


Gbr. 1 Rancangan topologi 1

Berikut spesifikasi yang digunakan dalam topologi :
- *core switch* : *ryu controller*
- *Aggregation switch* : 2
- *Edge switch* : 4
- *Host* : 8

ISSN (Online) 2747-0563
Volume 2 Tahun 2021

Berikut ini Gbr. 2 yang menunjukkan topologi 2 yang di gunakan



Gbr. 2 Rancangan topologi 2

Berikut spesifikasi yang digunakan dalam topologi 2 :
- *core switch* : *ryu controller*
- *Aggregation switch* : 3
- *Edge switch* : 6
- *Host* : 12

III. HASIL DAN PEMBAHASAN

A. Skenario Uji Coba

Skenario uji coba ini menggunakan *Iperf* dan *D-ITG* dengan beberapa jenis paket yaitu *UDP, DNS* dan *Telnet,* dengan parameter ujinya adalah rata-rata *delay* dan *rata-rata packet rate* dan pengujiannya dilakukan masing-masing sebanyak 10 kali.

a. Delay

TABEL I
DELAY TOPOLOGI 1

Percobaan	Delay (ms)		
	UDP Flow	DNS	Telnet
1	37,52	12,28	19,68
2	37,28	12,42	19,39
3	37,28	11,09	19,44
4	37,06	12,67	19,93
5	37,21	12,35	19,13
6	37,23	11,01	18,96
7	37,07	12,25	18,83
8	37,35	12,49	19,24
9	37,26	11,54	19,11
10	37,37	12,68	18,98

TABEL II
DELAY TOPOLOGI 2

Percobaan	Delay (ms)		
	UDP Flow	DNS	Telnet
1	41,69	11,86	20,61
2	41,31	12,47	19,75
3	41,61	11,96	20,55
4	41,63	11,46	19,23
5	41,35	12,02	19,33
6	41,39	11,98	20,41
7	41,27	11,49	19,23
8	41,49	11,20	19,72
9	40,51	12,07	19,70
10	40,42	12,45	18,66

Dari hasil uji coba pada tabel I dan table II untuk *delay* pada DNS pada topologi 1 lebih kecil daripada topologi 2 karena hambatan pada topologi 1 lebih sedikit di lihat dari jumlah switch sehingga *background* trafiknya lebih kecil.

b. *Packet rate*

TABEL III
PACKET RATE TOPOLOGI 1

Percobaan	Packet rate (Pkt/s)		
	UDP Flow	DNS	Telnet
1	87,79	0,5768	75,16
2	88,25	0,5867	73,11
3	88,29	0,5767	81,77
4	88,98	0,5766	78,15
5	88,55	0,5767	73,32
6	88,39	0,5767	71,31
7	88,76	0,5767	76,24
8	88,14	0,5768	77,49
9	88,38	0,5767	103,86
10	88,16	0,5766	102,14

TABEL IV
PACKET RATE TOPOLOGI 2

Percobaan	Packet rate (Pkt/s)		
	UDP Flow	DNS	Telnet
1	82,02	0,576	98,34
2	82,71	0,576	94,13
3	82,15	0,576	98,38
4	82,14	0,576	93,58
5	82,65	0,577	111,85
6	82,57	0,578	91,38
7	82,87	0,577	93,58
8	82,43	0,579	106,64
9	82,40	0,579	85,53
10	82,41	0,577	83,67

Dari hasil tabel III dan tabel IV untuk *packet rate* terbaik ada pada Telnet di topologi 2 karena beban trafik nya lebih stabil

IV. KESIMPULAN

Load balancing pada jaringan SDN dengan topologi *bipartite* pada beberapa jenis paket UDP *Flow*, DNS dan Telnet dengan parameter ujinya *delay* dan *packet rate* menunjukkan hasil sebagai berikut bahwa *packet rate* tergantung dengan *delay* yang didapatkan pada topologi yang ada, jika *delay* tinggi maka *packet rate* akan menurun. Dari hasil pengujian *delay* terkecil didapatkan oleh jenis paket DNS pada topologi 1 dengan *delay* 11,38 ms, *packet rate* terbesar didapatkan oleh Telnet pada topologi 2 dengan *packet rate* 92.07 pkt/s. Perbedaan dengan hasil penelitian sebelumnya adalah pada topologi yang digunakan yaitu topologi *bipartite* yang dipakai peneliti sedangakan terdahulu menggunakan topologi *fat-tree*. Skenario uji coba menggunakan 3 trafik jaringan yang berbeda yaitu UDP Flow, Telnet, dan DNS dengan mengambil parameter delay dan packet rate dengan setiap link mempunyai background traffic 100Mbit/s.

ISSN (Online) 2747-0563
Volume 2 Tahun 2021

REFERENSI

[1] Mustafa, M.E and Ibrahim, A.M.A. 2015. *Load Balancing Algorithms Round-Robin, Least-Connection and Least Loaded Efficiency*. Shaqra University: Shaqra, Saudi Arabia. ISSN 2279-0764.

[2] J. Moy, OSPF Version 2, document RFC 2328, IETF, 1998. [Online]. Available: http://www.ietf.org/rfc/rfc2328.txt

[3] Z. Cao, Z. Wang, and E. W. Zegura, "Performance of hashing-based schemes for internet load balancing," in Proc. IEEE INFOCOM, vol. 1. Mar. 2000, pp. 332–341.

[4] M. Chiesa, G. Kindler, and M. Schapira, "Trafic Engineering With Equal-Cost-Multipath: An Algorithmic Perspective," Proc. IEEE/ACM Transactions on Networking, vol.25, issue 2, pp.779-792, April 2017.

[5] K. Bakshi, "Considerations for Software Defined Networking (SDN): Approaches and use cases," in Proc. of IEEE Aerospace Conf., 2013 , pp. 1-9.

[6] S. Agarwal, M. Kodialam, and T. V. Lakshman, "Traffic engineering in software defined networks," in Proc. of IEEE INFOCOM, 2013, pp. 2211-2219.

[7] S. Fang, Y. Yu, C. H. Foh, and K. M. M. Aung, "A Loss-Free Multipathing Solution for Data Center Network Using Software-Defined Networking Approach," IEEE Trans. on Magnetics, vol. 49, no. 6, pp. 2723- 2730, 2013

[8] F. Hu, Q. Hao, and K. Bao, "A survey on software-defined network and OpenFlow: From concept to implementation," IEEE Communications Surveys & Tutorials, vol. 16, no. 4, pp. 2181-2206, 2014.

[9] I. F. Akyildiz, A. Lee, P. Wang, M. Luo, and W. Chou, "Research challenges for traffic engineering in software-defined networks," IEEE Netw., 2016

[10] McCauley, M., "POX", from http://www.noxrepo.org/, 2012

[11] Asadollahi, S., Gowsami, B., "Implementation of SDN using OpenDaylight Controller." Proceeding of An International Conference on Recent Trends in IT Innovations - Tec'afe 2017. ISSN(Online) : 2320-9801

[12] Project Floodlight, Floodlight. (2012). From http://floodlight.openflowhub.org/

[13] Erickson, D.,."The Beacon OpenFlow controller." Proceedings of ACM SIGCOMM Workshop Hot Topocs Software Defined Network II, 13-18 p, 2013.

[14] Nippon Telegraph and Telephone Corporation, RYU network operating system, 2012, from http://osrg.github.com/ryu

[15] Gude al, N., "NOX: Towards an operating system for networks." ACM SIGCOMM - Computer Communication Revie. vol. 38, no. 3, pp. 105–110.

[16] Asadollahi, S., Gowsami, B., "Software Defined Network, Controller Comparison." Proceedings of Tec'afe 2017,Vol.5, Special Issue 2, April 2017. ISSN: 2320-9798.

[17] McKeown et al, N., "OpenFlow: Enabling innovation in campus networks." ACM SIGCOMM - Computer Communication Revie, vol. 38, no. 2, p. 69–74, 2008..